CIT 591 Introduction to Software Development

Module 11: Version Control, Git & GitHub

Module Learning Objectives

- Keep track of changes in your project
- Revert to previous versions of code
- Collaborate with others while controlling versions
- Resolve conflicts that arise when your collaborator edits the same document as you
- Write your own constructors to construct instances of your class instead of relying on Java's default constructor

Module Glossary

- commit - a command that captures a snapshot of the repository's current changes
- repository - generally speaking, this is a project folder with some kind of version control enabled (in our case, git).
- log - display a log of all the commits and the corresponding commit messages captured across code development
- remote - To communicate with the rest of the world, git uses remotes. These are repositories other than the one on your computer to which you can send your changes. This has come to be associated with a repository in the cloud. GitHub is the most popular cloud-based Git repository hosting service.
- pull - pull down changes from the remote repository
- push - send changes to the remote repository

Module Resources

- **Textbook Readings:**
  - none
- **Websites:**
  - Git for Windows
  - Pull request tutorial
  - Git Cheat Sheet

- ProGit Book (Free)
- Resources to learn Git by reading and doing

Key Concepts & Examples

Git Basics:
Git is a popular version control system used in many software companies.
The general workflow is that if you make a change to a file, git add that file, git commit that file. Best practices: commit often and write descriptive commit messages.

For adding and then committing a file (remember, it is a 2 step process)

```
git add filename
git commit -m “commit message”
```

GitHub with Collaborators:
GitHub is a web-based hosting service for version control using Git.
`git pull --rebase` to get changes that your collaborators have made and put your changes on top.
`git push origin master` to push changes to the cloud.
If you and your collaborator both change a file simultaneously, there might be a conflict. Remember to resolve conflicts before pushing changes.

Creating Branches:
Create branches when you want to work on code without affecting the rest of your development team.
`git merge` is command you need to use to merge changes in your branch to the master branch or vice versa.